Software System for the Mars 2020 Mission Sampling and Caching Testbeds

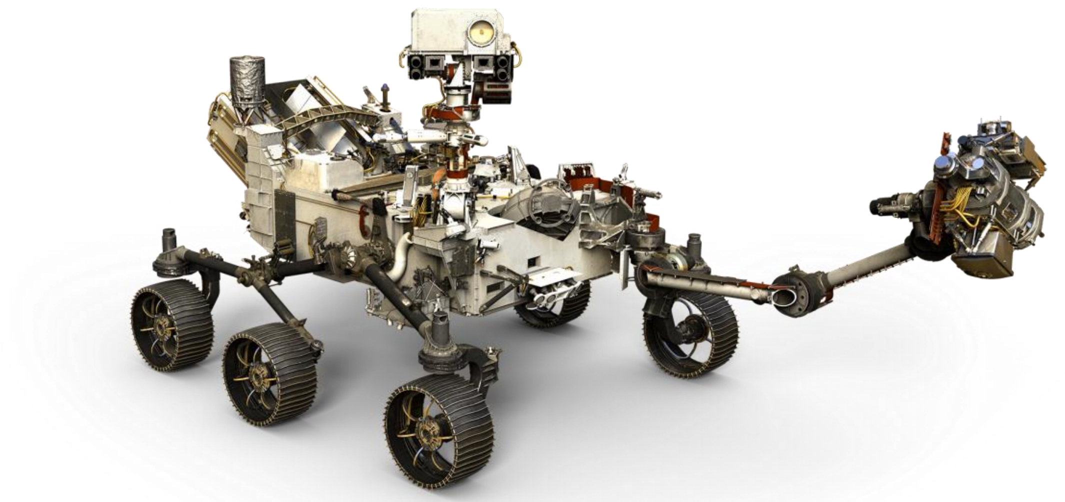
Kyle Edelberg, Paul Backes, Jeffrey Biesiadecki, Sawyer Brooks, Daniel Helmick, Won Kim, Todd Litwin, Brandon Metz, Jason Reid, Allen Sirota, Wyatt Ubellacker, Peter Vieira

Jet Propulsion Laboratory
California Institute of Technology

March 9, 2018

Background - Mars 2020 Rover Mission

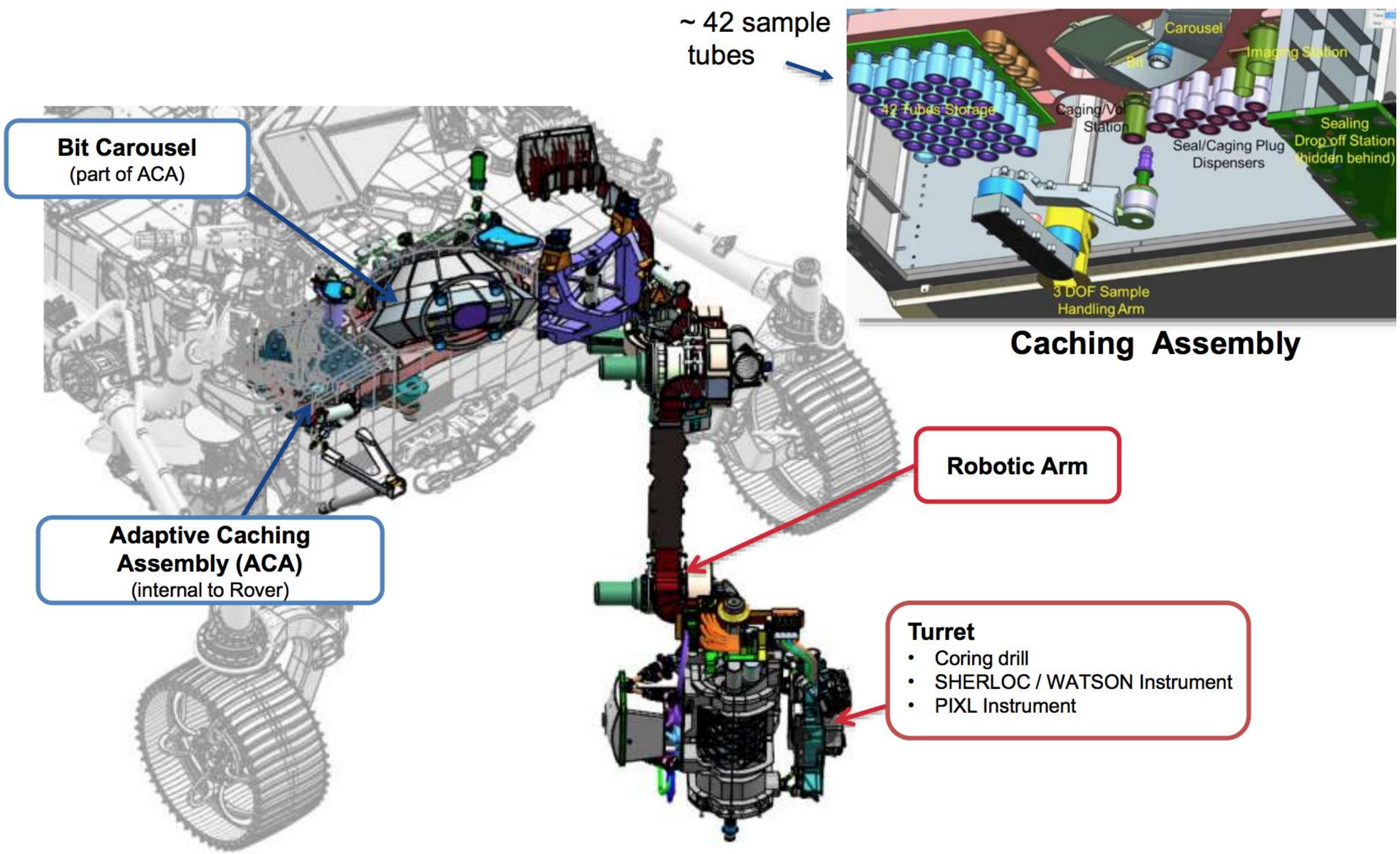
Objective: Address high-priority science goals for Mars exploration



https://mars.nasa.gov/mars2020/mission/overview/

- One objective is to collect sample cores and set them aside on the surface in a "cache" for potential subsequent retrieval
- This capability is to be achieved via the **Sampling and Caching Subsystem** (SCS), which is currently under development

Background: Sample Caching Subsystem (SCS)



http://sites.nationalacademies.org/cs/groups/ssbsite/documents/webpage/ssb_183291.pdf

Outline

1. Goals of this work
1. The CASAH software system
1. Supported testbeds
1. Hardware-software integration
2. Deployment, testing, and lessons learned
1. What next?

Testbed Software Objectives

1. Enable repeatable, **high throughput testing** of prototype hardware across a suite of testbeds.

1. Provide a light-weight framework for rapid development of **relevant algorithms** that can be tested on real hardware and inform flight software development

1. Allow for **non-developers to operate** the testbeds.

1. Collect **data products** that can be easily parsed and archived.

1. Provide robust **fault protection** to avoid hardware damage.

Software Solution

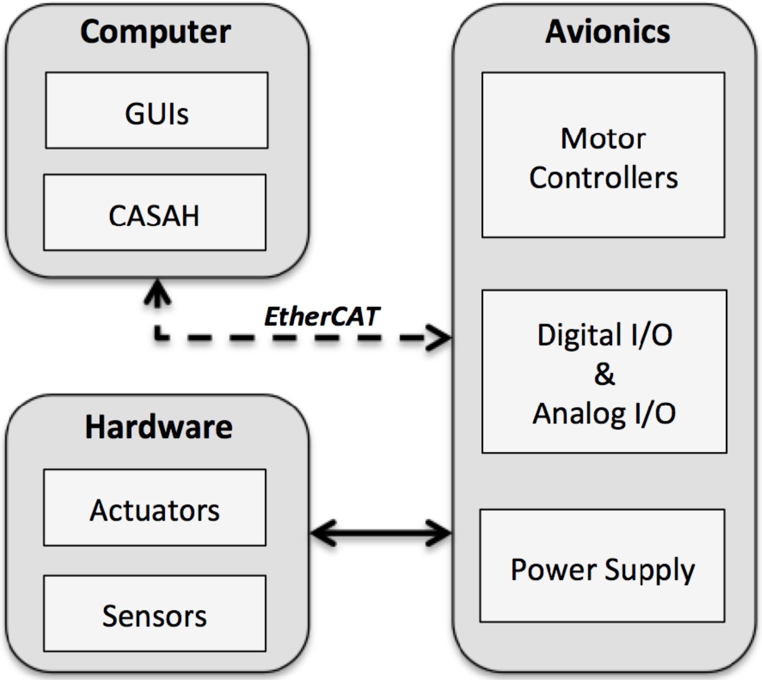
- Why not just use JPL Rover Flight Software (FSW)?
  - Long development timeline due to rigorous FSW standards
  - Scope does not encompass all testbed objectives
  - Operational complexity is high

- Why not use a commercial solution like LabVIEW?
  - LabVIEW is designed to operate a wide variety of hardware at a low level
  - It does not scale well to high-level algorithms and system level capabilities
  - Many components are black-boxes
  - It is not relevant to FSW because it is so fundamentally different

- Our solution: Controls and Autonomy for Sample Acquisition and Handling (CASAH)
  - Implementation of the Intelligent Robotic Systems Architecture (IRSA)
  - IRSA mimics JPL Rover Flight Software in the following ways:
    1. System is divided into modules, which communicate via message passing
    2. Each module is 'owned' by a single developer
    3. Operator interface to each module is explicitly defined by a command dictionary
  - CASAH primarily coded in C language, which is the same as FSW

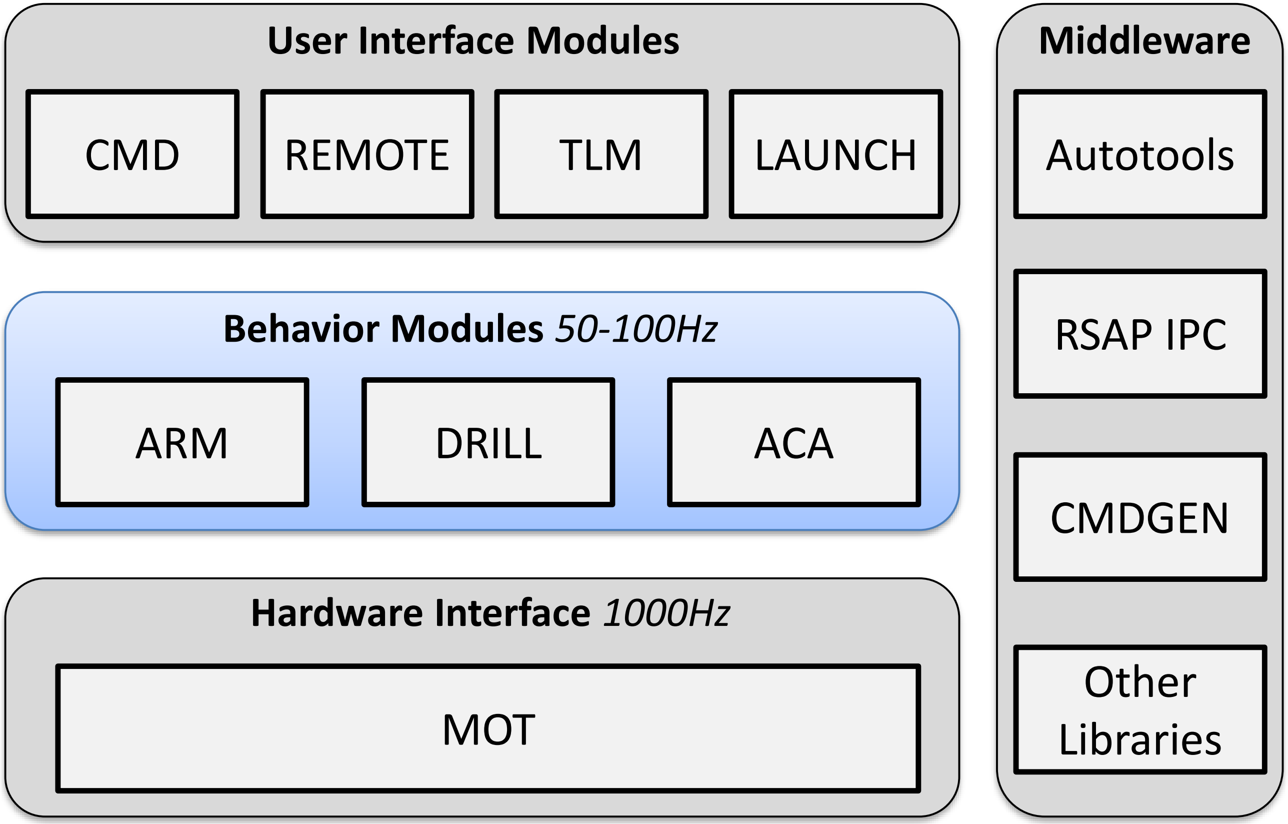
Controls and Autonomy for Sample Acquisition and Handling (CASAH)

- Development began in January 2014 by a team of three
  - First deployment in May 2014
- Development has continued to present day, with team size growing to roughly five
  - To date, CASAH has supported **10 testbeds** and **1400+ tests**
  - Test data has provided invaluable feedback to hardware and software teams
  - Flight-relevant algorithms have been developed, iterated, and tested



Context Diagram for CASAH

CASAH Components

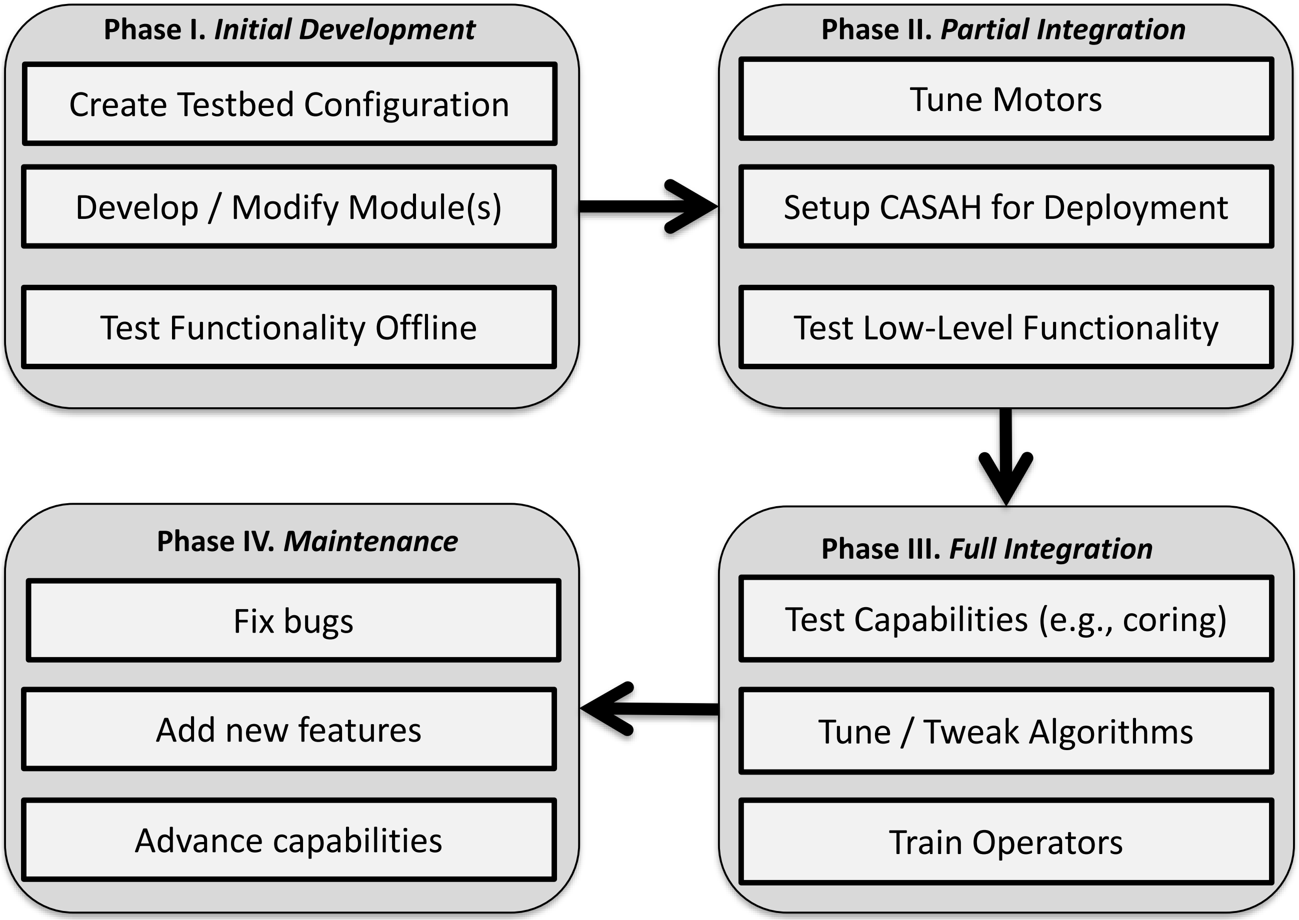


Supporting Testbeds with CASAH

- 10 testbeds, each with different combination of SCS components and distinct objectives
- How do we develop CASAH to support different testbeds?
  - CASAH is a single repository
  - Modules have testbed configuration files
  - Testbed is specified at build time

Top left: Arm and drill (ambient)
Top right: Arm and drill (thermal/vacuum)
Bottom left: Arm docking
Bottom right: ACA end effector

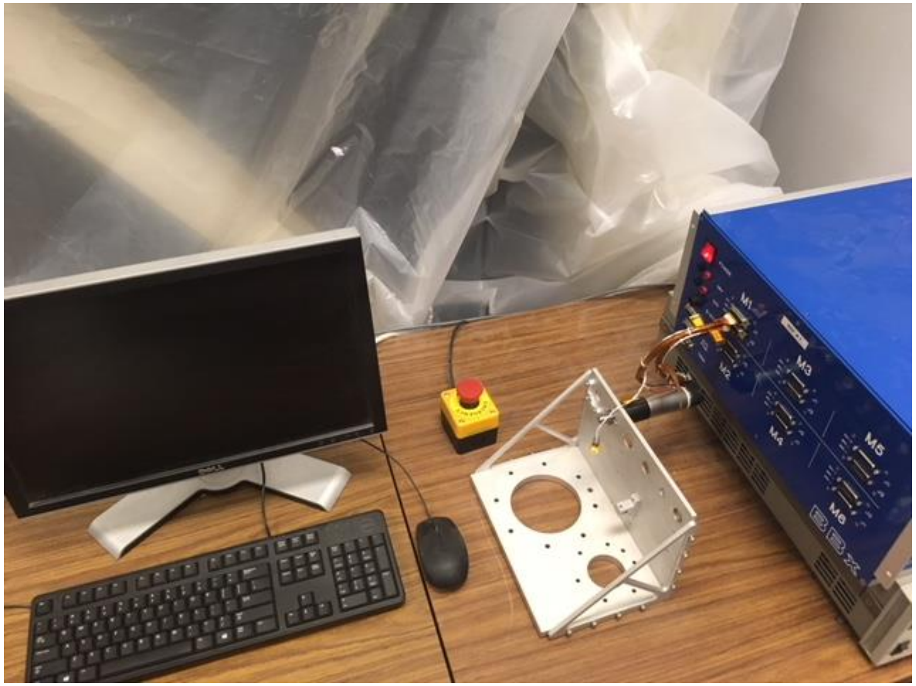
CASAH Development and Hardware-Software Integration



- This process grew organically
- Tight coupling of software-hardware teams before and during this process critical for success

Hardware-Software Integration: A Note About Avionics

- CASAH supports only EtherCAT devices
- Avionics are standardized across testbeds (BlueBox)
- Resulting software is maintainable and system debugging is simplified



CASAH development station

What is in a blue box?

- COTS motor controllers (Elmo Gold Whistles)
- COTS I/O modules (Beckhoffs)
- Power supply
- USB hub

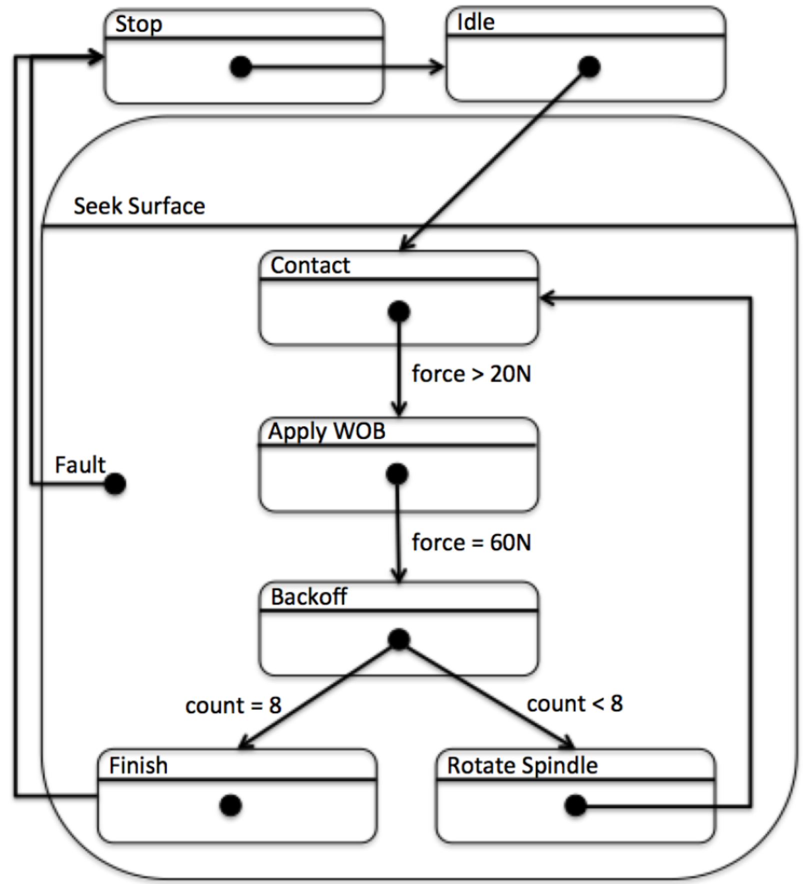
Results – Testing Throughput

Testbed	Hardware Components	Period	Number of Tests
1) Boundary Conditions Testbed	Robotic Arm, Coring Drill	08/2014-03/2015	93
2) Percussion Efficacy and Comminution	Coring Drill	02/2015-Active	426
3) Environmental Development Testbed	Robotic Arm, Coring Drill	04/2015-Active	282
4) Ambient Robotic Coring	Robotic Arm, Coring Drill	07/2015-05/2016	136
5) Tube Manipulation Testbed	ACA (Linear actuator only)	08/2015-08/2016	105
6) Docking Testbed	Robotic Arm	06/2016-Active	41
7) Surface Prep Operations Testbed	Coring drill, gas Dust Removal Tool	11/2016-Active	85
8) Single Station Testbed	ACA (Linear actuator and end-effector only)	03/2017-Active	159
9) Multi Station Testbed	ACA (no bit carousel or holder)	04/2017-Active	21
10) Percussion Mechanism Testbed	Coring drill (percussion only)	08/2017-Active	45

Number of tests ran on each SCS testbed.
Resulting data products have proved invaluable.

Results – Algorithm Transfer to FSW

- ACA, ARM, and DRILL modules are now under development in FSW
- Algorithms developed and tested in CASAH are being re-written into FSW
- Example: DRILL Seek Surface
  - Concept inspired by results seen during testing
  - Algorithm rapidly implemented and rolled into testing with CASAH on all testbeds with coring drills
  - With proven maturity, algorithm is now being coded into the DRILL module in FSW



Example algorithm: DRILL_SEEK_SURFACE; Developed in CASAH, currently being coded into Mars 2020 FSW

… But Nothing is Perfect

CASAH Deployment: Select issues and resolutions

Date	Issue	Resolution
08/2014	MOT timer slip during force control	Found that when DRILL was running force-control, MOT would often overrun its loop timer. Found that MOT was printing continuously at each request it received from DRILL at 100Hz. Modified MOT's logic to only print on first motion request.
03/2015	Some uncontrolled motion would occur when E-stop pressed	This affected a robotic arm for a testbed. If moving when the E-stop was pressed, the arm would fall under gravity a small amount before brakes closed. Determined to be a hardware limitation, but required adding a 'soft' E-stop feature, whereby a physical E-stop would toggle a digital input that CASAH would read and use to initiate a smooth motion ramp-down.
06/2015	Separate branches for each testbed infeasible for developers	Created testbed configuration files and integrated with build process. Changed CASAH to consist of single, unified master branch.
07/2015	MOT timer slip during graphics processing	Found that Nvidia graphics driver was clashing with MOT process at the OS level. Selected alternate graphics card with specific open-source driver that eliminated the clash. Retrofitted all computers with this card, and to ensure correct driver version started the master hard drive cloning system.
07/2015	Externals versions not being managed or tracked	Moved all externals from SVN to git on JPL's GitHub. Changed CASAH to pull in specific tag numbers of all externals via a version-controlled script.
08/2015	Hard drives getting full on operation computers	Added notification to alert operator that hard disk is getting full. Dropped MOT's data logging rate for MOT and behavior modules to 1Hz when no motors have been active for more than 60 seconds.
10/2015	Could not use Beckhoff module that had digital inputs and outputs	A Beckhoff module that had both inputs and outputs did not work with our EtherCAT drivers. Found bug in driver design, requiring major overhaul. Updated driver stack, performed significant testing, then switched CASAH to support new design.
11/2015	Still getting timer slips in MOT process	By tracing where timer slip was occurring, found fflush in several parts of low-level message printing. Added option to disable fflush, setting default to disable it for all CASAH modules.
05/2016	MOT would not shut down	Determined cause was persistent fault on the motor controller inhibiting MOT's state machine from allowing it to terminate. Added persistence counter and modified MOT's shutdown logic accordingly.
08/2016	Testbed operations computer kept locking up	Found that multiple instantiations of telemetry display were running. Found that CASAH script used to start the display was not checking if instance was already running in the background, which could happen if not closed properly. Updated script to alert operator if display is already running when they try to start it.

Summary and Lessons Learned

- Direct porting of code from testbed to FSW would impose harsh constraints on testbeds; direct porting of algorithms is a win-win
- Mimicking FSW architecture by implementing IRSA maintains relevancy
  - Behavior modules in CASAH are owned by the same people writing them in FSW, and functionality is 1:1 mapping between the two code bases
  - Command dictionary and sequencing enables repeatable, high throughput testing. Generated data products are essential for system development.
  - Message passing between modules works effectively for time-critical applications
- Early and continuous hardware-software integration is critical for system of this complexity
- Keeping code clean and lightweight pays off:
  - Explicitly restrict supported hardware
  - Re-write application level code often
  - Avoid the temptation to chase the 'perfect' software system which never needs modification

Please refer to paper for full list of specific, technical lessons learned

Looking Ahead

- For Mars 2020 SCS S/W development, focus is now Flight Software
  - We have developed a system which enables us to directly test SCS FSW modules with Blue Boxes; allows for direct comparison to CASAH performance
  - CASAH itself is now in maintenance phase
- For testbeds on new projects
  - Continue to implement IRSA architecture
  - Bulk of CASAH middleware will likely be reused, but application code rewritten
  - Lessons learned from CASAH have been captured and will inform new implementation
  - We continue to monitor the Robot Operating System (ROS) 2.0 for potential viability

Thank you

Questions?

Jet Propulsion Laboratory
California Institute of Technology

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.